Building forms for an engaging UX in VueJS

Divya Sasidharan

Divya Sasidharan

Developer Advocate @Netlify

@shortdiv

We're hiring!

netlify

https://boards.greenhouse.io/netlify

09:09

Share To

Share

Write a caption...

Add Location

Chicago, Illinois | Wicker Park | Chicago, Illinois | Wicker

Tag People

Share To

Facebook

Twitter

Tumblr

Advanced Settings

Your location

Choose destination, or click on the map..

Your location

Home SET LOCATION

Work SET LOCATION

Ponce City Market

Chai Pani West Ponce de Leon Avenue, Decatur, GA

Sheraton Suites Galleria-Atlanta Cobb Parkway Southea...

Big Chow Grill Galleria Parkway #1B1, Atlanta, GA

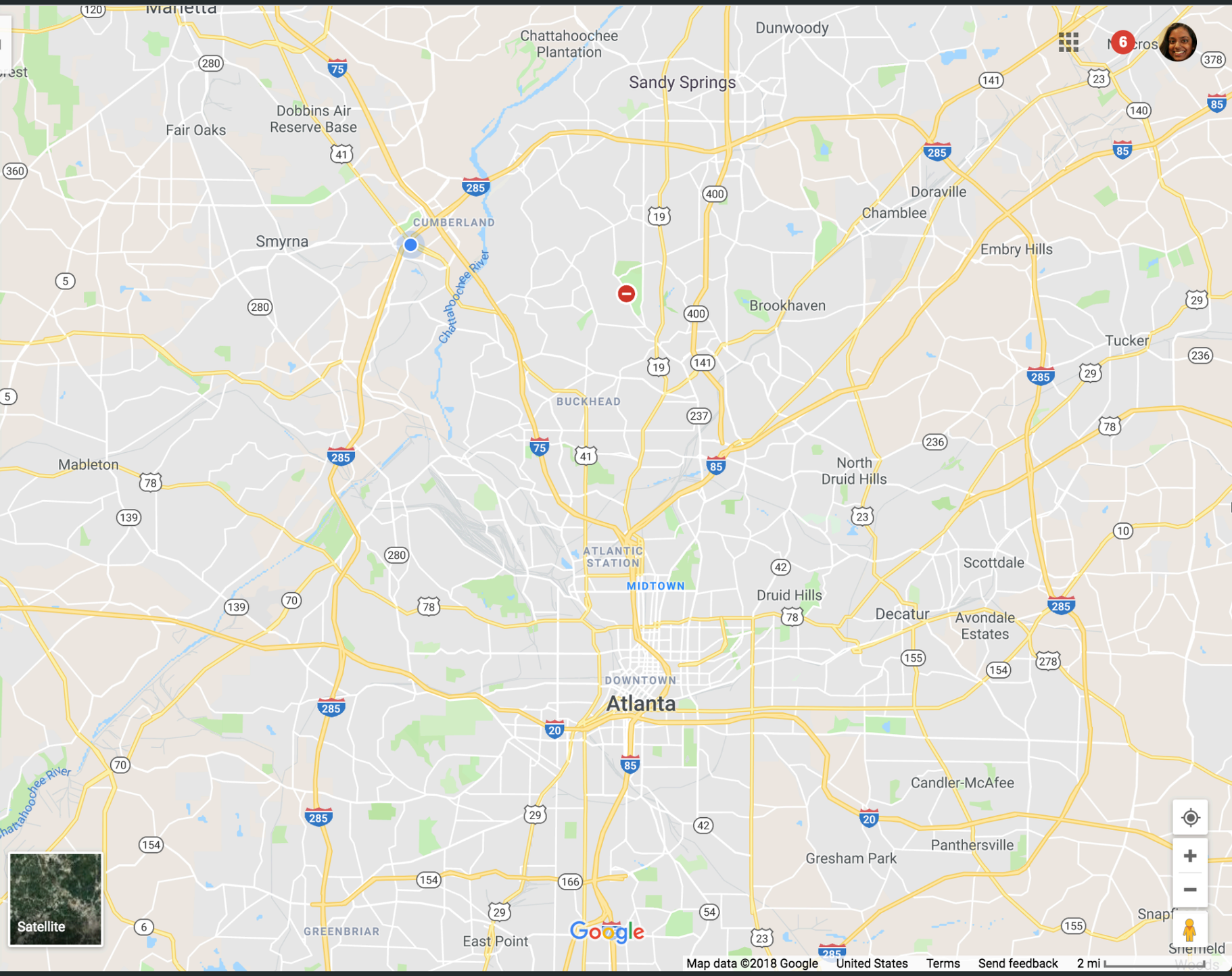
DELAYS

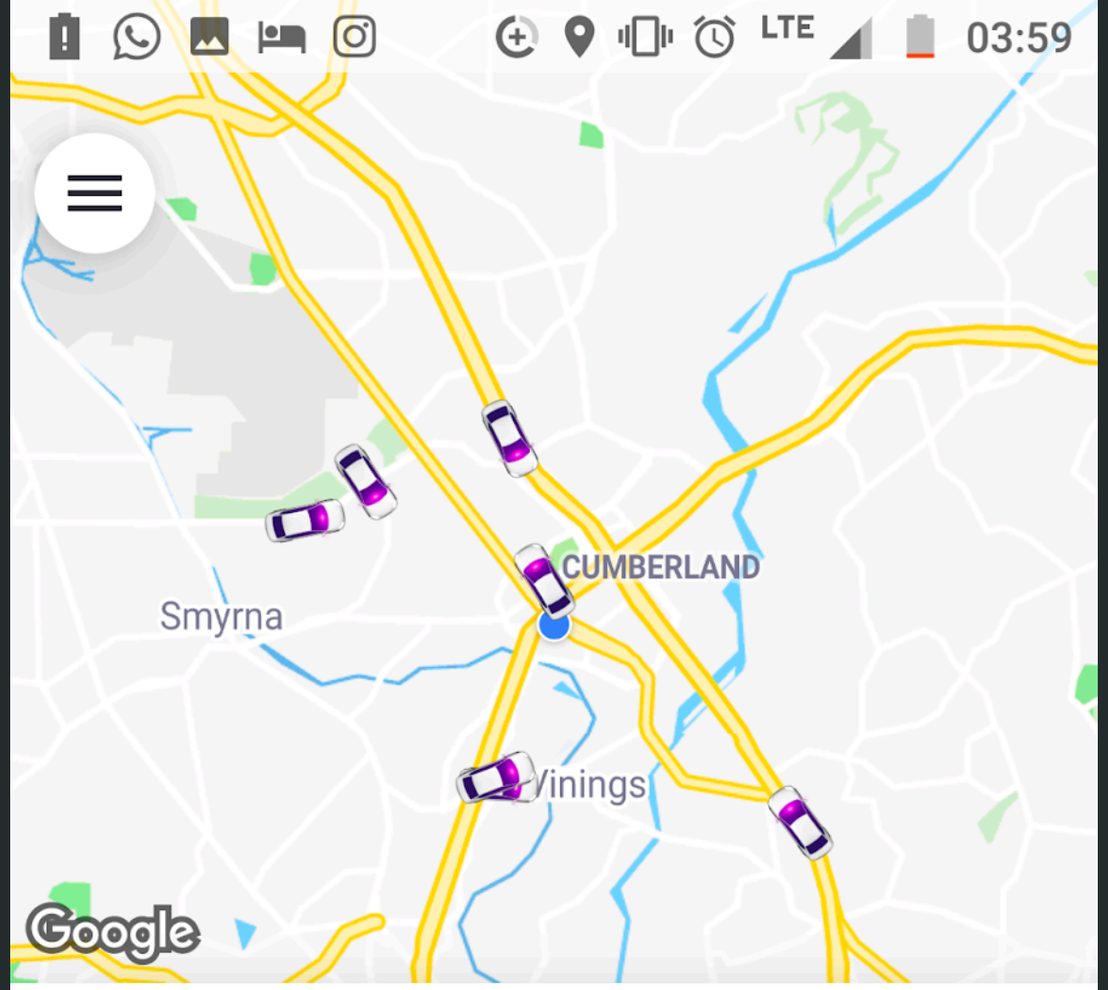
Light traffic in this area

Road closure on Macedonia Rd

Road closure on Powers Ferry Rd NW

Road closure on Powers Ferry Rd NW





Hey there, Divya

Where are you going?

Search destination

Hartsfield - Jackson Atlanta Inter...
Hartsfield - Jackson Atlanta Internatio...

Home

Cash Rules Everything Around Me

Forms

~~Cash~~ Rule Everything Around Me

"

Forms Suck.

We should design accordingly.

"

– *Luke Wroblewski*

How Users Read on the Web

turbotax Deluxe

Tax Home
Previous Taxes

2017 TAXES

My Info
Federal
State
Review
File

Upgrade
Tax Tools

Sign Out

We're almost there...

Uncommon tax situations

These situations are pretty uncommon. Let's keep going if none of these apply to you.

All uncommon situations

Alternative Minimum Tax — Show more

Business Taxes and Deductions — Show more
Business taxes and uncommon business deductions

Additional Tax Payments — Show more
Penalties, early retirement withdrawals, nanny tax, apply refund to next year

What makes a form “good”?

A Good Form is Simple
A Good Form is Doing Stuff
A Good Form is Talkative
A Good Form is Unbreakable
A Good Form is Fun
A Good Form is Concise
A Good Form is Well Formatted
A Good Form is Easy to Scan
A Good Form is Easy to Navigate

A Good Form is Simple
A Good Form is Doing Stuff
A Good Form is Talkative
A Good Form is Unbreakable
A Good Form is Fun
A Good Form is Concise
A Good Form is Well Formatted
A Good Form is Easy to Scan
A Good Form is Easy to Navigate

A Good Form is **Doing Stuff**

Tea Fog

- Earl Grey
- Matcha
- Rooibos
- Yerba Mate

Tea Fog

- Earl Grey
- Matcha
- Rooibos
- Yerba Mate

```
<input
  type="radio"
  name="teaType"
  :value=""
  :checked=""
 @change=""
>
```

```
<input
  type="radio"
  name="teaType"
  :value=""
  :checked=""
  @change=""
>
```

```
<input
  type="radio"
  name="teaType"
  :value=""
  :checked=""
  @change=""
>
```

```
<input
  type="radio"
  name="teaType"
  :value=""
  :checked=""
 @change=""
>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked=“chosenTea === tea"
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
   </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [“Earl Grey", "Matcha", “Rooibos”, “Yerba Mate"],
      chosenTea: “Earl Grey”
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked=“chosenTea === tea"
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
    </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [“Earl Grey", "Matcha", “Rooibos”, “Yerba Mate"],
      chosenTea: “Earl Grey”
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked=“chosenTea === tea"
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
    </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [ “Earl Grey", "Matcha", “Rooibos”, “Yerba Mate” ],
      chosenTea: “Earl Grey”
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked=“chosenTea === tea"
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
    </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [ “Earl Grey", "Matcha", “Rooibos”, “Yerba Mate” ],
      chosenTea: “Earl Grey”
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked=“chosenTea === tea”
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
    </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [ “Earl Grey", "Matcha", “Rooibos”, “Yerba Mate” ],
      chosenTea: “Earl Grey”
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          :checked="chosenTea === tea"
          @input="ev => chosenTea = ev.target.value">
          <span>{{ tea }}</span>
      </label>
   </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [“Earl Grey", "Matcha", “Rooibos”, “Yerba Mate"],
      chosenTea: "Earl Grey"
    };
  }
}
</script>
```

```
<template>
  <div>
    <form name=“vue-forms” method=“post”
          @submit.prevent=“handleSubmit">
      <label v-for="tea in teaTypes">
        <input
          type="radio"
          name="tea"
          :value="tea"
          v-model="chosenTea"
          >
          <span>{{ tea }}</span>
      </label>
   </form>
  </div>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      teaTypes: [“Earl Grey", "Matcha", “Rooibos”, “Yerba Mate"],
      chosenTea: "Earl Grey"
    };
  }
}
</script>
```

Demo

A Good Form is **Talkative**

Tea

- (•) Earl Grey
- ( ) Irish Breakfast
- ( ) Chai
- ( ) Rose
- ( ) Matcha
- ( ) Rooibos
- ( ) Yerba Mate
- ( ) Green

Milk

Tea

- [x] Earl Grey
- [ ] Irish Breakfast
- [ ] Chai
- [ ] Matcha

Milk

- [ ] Whole Milk
- [x] Soy Milk

Order Up!

x

1 Manchester Fog

Coming Right Up!

```
<template>
  <div class=“notification” v-if="hasNotifications">
    …
  </div>
</template>

<script>
export default {
  name: "notification",
  props: {
    status: {
      type: Object
    },
  },
  data () {
    return {
      hasNotification: false,
      countdown: null,
    }
  },
  watch: {
    status () {
      this.hasNotifications = true;
      this.countdown = setTimeout(this.removeNotification, 3000)
    }
  },
  methods: {
    removeNotification () {
      clearTimeout(this.countdown)
      this.hasNotification = false
    }
  }
}
</script>
```

```
<template>
  <div class=“notification” v-if="hasNotifications">
    …
  </div>
</template>

<script>
export default {
  name: "notification",
  props: {
    status: {
      type: Object
    },
  },
  data () {
    return {
      hasNotification: false,
      countdown: null,
    }
  },
  watch: {
    status () {
      this.hasNotifications = true;
      this.countdown = setTimeout(this.removeNotification, 3000)
    }
  }
  methods: {
    removeNotification () {
      clearTimeout(this.countdown)
      this.hasNotification = false
    }
  }
}
</script>
```

```
<template>
  <div class=“notification” v-if=“hasNotifications">
    …
  </div>
</template>

<script>
export default {
  name: “notification",
  props: {
    status: {
      type: Object
    },
  },
  data () {
    return {
      hasNotification: false,
      countdown: null,
    }
  },
  watch: {
    status () {
      this.hasNotifications = true;
      this.countdown = setTimeout(this.removeNotification, 3000)
    }
  }
  methods: {
    removeNotification () {
      clearTimeout(this.countdown)
      this.hasNotification = false
    }
} }
</script>
```

```
<template>
  <div class="notification" v-if="hasNotifications">
    …
  </div>
</template>

<script>
export default {
  name: "notification",
  props: {
    status: {
      type: Object
    },
  },
  data () {
    return {
      hasNotification: false,
      countdown: null,
    }
  },
  watch: {
    status () {
      this.hasNotifications = true;
      this.countdown = setTimeout(this.removeNotification, 3000)
    }
  },
  methods: {
    removeNotification () {
      clearTimeout(this.countdown)
      this.hasNotification = false
    }
  }
}
</script>
```

```
<template>
  <div class=“notification”>v-if="hasNotifications">
    …
  </div>
</template>

<script>
export default {
  name: "notification",
  props: {
    status: {
      type: Object
    },
  },
  data () {
    return {
      hasNotification: false,
      countdown: null,
    }
  },
  watch: {
    status () {
      this.hasNotifications = true;
      this.countdown = setTimeout(this.removeNotification, 3000)
    }
  },
  methods: {
    removeNotification () {
      clearTimeout(this.countdown)
      this.hasNotification = false
    }
  }
}
</script>
```

Demo

Milk

- (•) Whole Milk
- ( ) Soy Milk

Milk i

Anyone who has used that comforting phrase ‘a nice cup of tea’ invariably means Indian tea.

- Whole Milk
- Soy Milk

Demo

A Good Form is **Unbreakable**

London Fog

- (•) Earl Grey
- ( ) Irish Breakfast
- ( ) Chai
- ( ) Rose
- ( ) Matcha
- ( ) Rooibos
- ( ) Yerba Mate
- ( ) Green

- (•) Whole Milk
- ( ) Soy Milk

```
<template>
  …
  <label
    v-for="milk in milkNames">
    <input
      name="milk"
      type="radio"
      :disabled=“isMilkDisabled(milk)"
      …>
    <span>{{ milk }}</span>
  </label>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      milkNames: [‘Whole Milk’, ‘Soy Milk’]
    }
  },
  methods: {
    isMilkDisabled (milk) {
      // check that there is the available milk options based on a selected Tea
    }
  },
}
</script>
```

```
<template>
  …
  <label
    v-for="milk in milkNames">
    <input
      name="milk"
      type="radio"
      :disabled=“isMilkDisabled(milk)"
      …>
    <span>{{ milk }}</span>
  </label>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      milkNames: [‘Whole Milk’, ‘Soy Milk’]
    }
  },
  methods: {
    isMilkDisabled (milk) {
      // check that there is the available milk options based on a selected Tea
    }
  },
}
</script>
```

```
<template>
  …
  <label
    v-for="milk in milkNames">
    <input
      name="milk"
      type=“radio"
      :disabled=“isMilkDisabled(milk)"
      …>
    <span>{{ milk }}</span>
  </label>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      milkNames: [‘Whole Milk’, ‘Soy Milk’]
    }
  },
  methods: {
    isMilkDisabled (milk) {
      // check that there is the available milk options based on a selected Tea
    }
  },
}
</script>
```

```
<template>
  …
  <label
    v-for="milk in milkNames">
    <input
      name="milk"
      type=“radio"
      :disabled=“isMilkDisabled(milk)”
      …>
    <span>{{ milk }}</span>
  </label>
</template>

<script>
export default {
  name: "TeaFog",
  data () {
    return {
      milkNames: [‘Whole Milk’, ‘Soy Milk’]
    }
  },
  methods: {
    isMilkDisabled (milk) {
      // check that there is the available milk options based on a selected Tea
    }
  },
}
</script>
```

```
teaTypes: {
  "Earl Grey": ['London', 'Manchester', 'Seattle'],
  "Irish Breakfast": ['Dublin'],
  “Chai":['Bombay'],
  "Rose": ['Atlantic City'],
  "Matcha": ['Tokyo'],
  "Rooibos": ['Cape Town'],
  "Yerba Mate":['Montreal'],
  "Green": ['Oregon Mist']
},
milkTypes: {
  'Whole Milk':['London', 'Oregon Mist', 'Dublin', …],
  'Soy Milk’: ['Manchester', 'Seattle'],
}
```

```
teaTypes: {
  "Earl Grey": ['London', 'Manchester', 'Seattle'],
  "Irish Breakfast": ['Dublin'],
  “Chai":['Bombay'],
  "Rose": ['Atlantic City'],
  "Matcha": ['Tokyo'],
  "Rooibos": ['Cape Town'],
  "Yerba Mate":['Montreal'],
  "Green": ['Oregon Mist']
},
milkTypes: {
  'Whole Milk':['London', 'Oregon Mist', 'Dublin', …],
  'Soy Milk’: ['Manchester', 'Seattle'],
}
```

```
teaTypes: {
  "Earl Grey": ['London', 'Manchester', 'Seattle'],
  "Irish Breakfast": ['Dublin'],
  “Chai":['Bombay'],
  "Rose": ['Atlantic City'],
  "Matcha": ['Tokyo'],
  "Rooibos": ['Cape Town'],
  "Yerba Mate":['Montreal'],
  "Green": ['Oregon Mist']
},
milkTypes: {
  'Whole Milk':['London', 'Oregon Mist', 'Dublin', …],
  'Soy Milk’: ['Manchester', 'Seattle'],
}
```

```
teaTypes: {
  "Earl Grey": ['London', 'Manchester', 'Seattle'],
  "Irish Breakfast": ['Dublin'],
  “Chai":['Bombay'],
  "Rose": ['Atlantic City'],
  "Matcha": ['Tokyo'],
  "Rooibos": ['Cape Town'],
  "Yerba Mate":['Montreal'],
  "Green": ['Oregon Mist']
},
milkTypes: {
  'Whole Milk':['London', 'Oregon Mist', 'Dublin', …],
  'Soy Milk’: ['Manchester', 'Seattle'],
}
```

Demo

Contact Info

Phone Number

(XXX)-XXX-XXXX

```
<template>
  <div>
    <label>
      <input
        type=“tel"
        :value="formatNumber"
        @change="handleChange"
        @keyup="handleChange"
      >
    </label>
  </div>
</template>

</script>
import { NorthAmericanize } from "../utils/lib.js";

export default {
  name: "PhoneField",
  data() {
    return {
      formatNumber: ""
    };
  },
  methods: {
    …
    handleChange(e) {
      if (this.isNotValidNumChar(e.which) && this.isNotDelete(e.which)) {
        e.preventDefault();
        return;
      }
      this.number = NorthAmericanize(e, e.target.value);
    }
  }
};
</script>
```

```
<template>
  <div>
    <label>
      <input
        type=“tel"
        :value="formatNumber"
        @change="handleChange"
        @keyup="handleChange"
      >
    </label>
  </div>
</template>

</script>
import { NorthAmericanize } from "../utils/lib.js";

export default {
  name: "PhoneField",
  data() {
    return {
      formatNumber: ""
    };
  },
  methods: {
    …
    handleChange(e) {
      if (this.isNotValidNumChar(e.which) && this.isNotDelete(e.which)) {
        e.preventDefault();
        return;
      }
      this.number = NorthAmericanize(e, e.target.value);
    }
  }
};
</script>
```

```
<template>
  <div>
    <label>
      <input
        type=“tel"
        :value="formatNumber"
        @change="handleChange"
        @keyup="handleChange"
      >
    </label>
  </div>
</template>

</script>
import { NorthAmericanize } from "../utils/lib.js";

export default {
  name: "PhoneField",
  data() {
    return {
      formatNumber: ""
    };
  },
  methods: {
    …
    handleChange(e) {
      if (this.isNotValidNumChar(e.which) && this.isNotDelete(e.which)) {
        e.preventDefault();
        return;
      }
      this.number = NorthAmericanize(e, e.target.value);
    }
  }
};
</script>
```

```
<template>
  <div>
    <label>
      <input
        type=“tel"
        :value="formatNumber"
        @change="handleChange"
        @keyup="handleChange"
      >
    </label>
  </div>
</template>

</script>
import { NorthAmericanize } from "../utils/lib.js";

export default {
  name: "PhoneField",
  data() {
    return {
      formatNumber: ""
    };
  },
  methods: {
    …
    handleChange(e) {
      if (this.isNotValidNumChar(e.which) && this.isNotDelete(e.which)) {
        e.preventDefault();
        return;
      }
      this.number = NorthAmericanize(e, e.target.value);
    }
  }
};
</script>
```

Demo

A Good Form is **Fun to Complete**

Email

email@domain.com

ewwe

Log in

Demo

Which did you hear?

- Yanny
- Laurel
- Both

Just tell me already

Demo

Netlify → firebase

↑

Actions

Component

Mutations

State

A Good Form is **Fun to Work On**


netlify

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    name=“vue-tea-form"
    method=“post"
    data-netlify=“true”
    data-netlify-honeypot=“bot-field"
    @submit.prevent="handleSubmit">
    <input
    type=“hidden"
    name=“form-name"
    value=“vue-tea-full-form“ />
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit() {…}
  },
}
</script>
```

```
<template>
  <form
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit () {
      fetch('/', {
        method: 'POST',
        headers: {
          "Content-Type": “application/x-www-form-urlencoded"
        },
        body: this.encode({
          'form-name': ‘vue-tea-full-form’,
          ... this.form
        })
      })
    }
  }
}
</script>
```

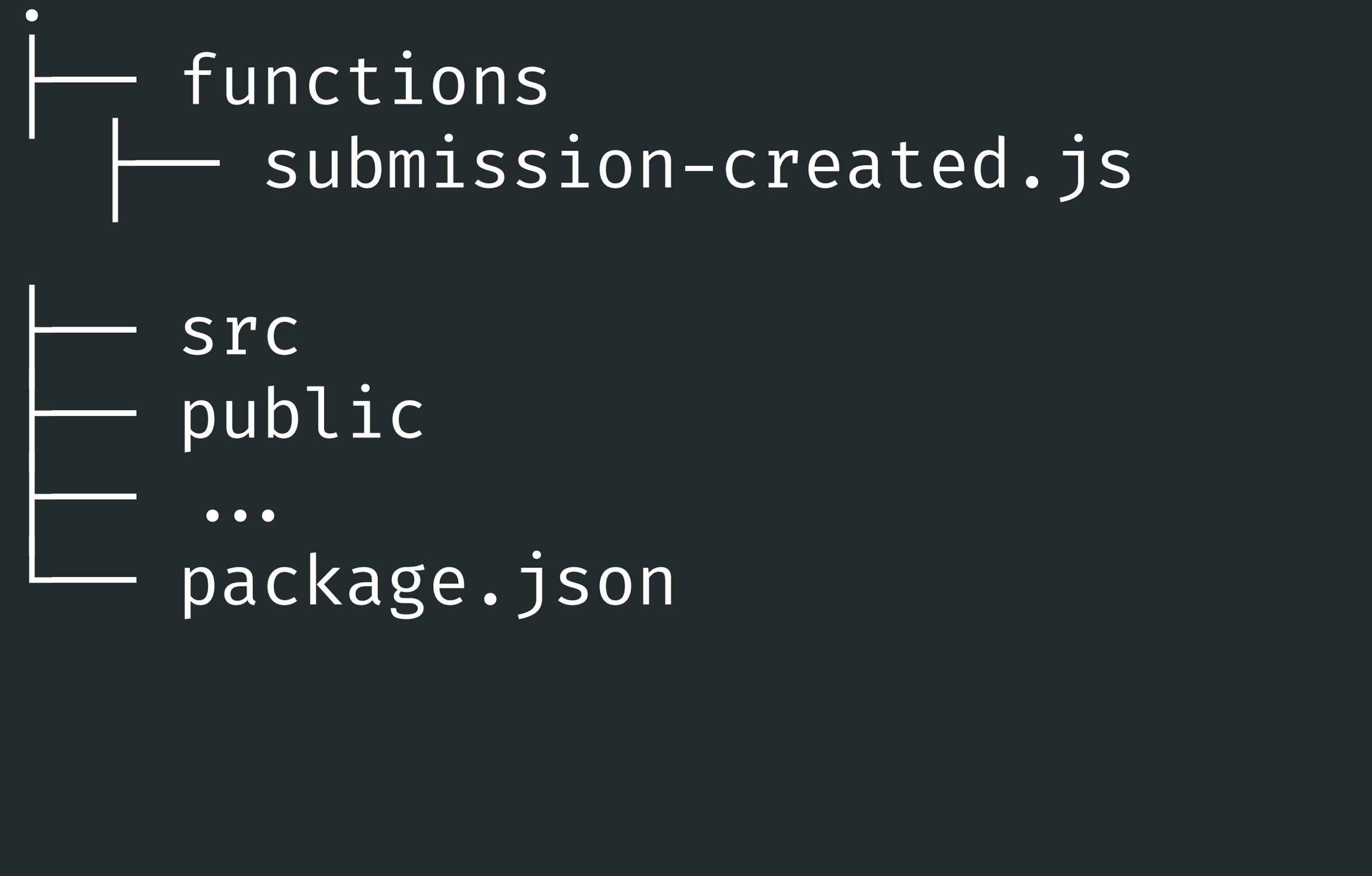
```
<template>
  <form
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit () {
      fetch('/', {
        method: 'POST',
        headers: {
          "Content-Type": “application/x-www-form-urlencoded"
        },
        body: this.encode({
          'form-name': ‘vue-tea-full-form’,
          ... this.form
        })
      })
    }
  }
}
</script>
```

```
<template>
  <form
    …
  </form>
</template>
<script>
export default {
  name: "Form",
  methods: {
    handleSubmit () {
      fetch('/', {
        …
      })
      .then(() ⇒ {
        this.$router.push(‘thanks')
      })
      .catch(err ⇒ {
        this.$router.push('404')
      });
    }
  }
}
</script>
```

```
import Vue from 'vue';
import Router from 'vue-router';
import Form from '../components/Form';
import SubmissionSuccess from '../components/SubmissionSuccess'
import SubmissionFail from '../components/SubmissionFail'

Vue.use(Router);

const router = new Router({
  routes: [
    {
      path: '/',
      name: 'Form',
      component: QAForm
    },
    {
      path: '/thanks',
      name: 'success',
      component: SubmissionSuccess
    },
    {
      path: '/404',
      name: 'fail',
      component: SubmissionFail
    }
  ]
})

export default router;
```

```
const path = require('path')
const PrerenderSPAPlugin = require('prerender-spa-plugin')

module.exports = {
  configureWebpack: () => {
    if (process.env.NODE_ENV !== 'production') return;
    return {
      plugins: [
        new PrerenderSPAPlugin(
          // Absolute path to compiled SPA
          path.resolve(__dirname, 'dist'),
          // List of routes to prerender
          [ '/'],
          {
            // options
          }
        ),
      ]
    }
  }
}
```

vue.config.js


functions
submission-created.js
src
public
...
package.json

```
exports.handler = function(event, context, callback)
{
    // your server-side magic ✨✨✨🧙✨✨✨ //
}
```

Forms

Forms Free

1 form collecting data.

Last submission received an hour ago.

Learn more about form handling in the docs →

Settings and usage

Active forms

yanny-v-laurel

Last submission at 3:50 AM (an hour ago)

71 submissions ›

Submissions from yanny-v-laurel

71 submissions.

Last submission at 3:50 AM (an hour ago)

Learn more about form handling in the docs →

Download as CSV

Showing 1-20 of 71 submissions

Expand all

Laurel	3:50 AM
Yanny	3:50 AM
Yanny	3:50 AM
Laurel	3:49 AM
Both	2:02 AM
Laurel	Oct 17
Yanny	Oct 17

A Good Form is Simple
A Good Form is Doing Stuff
A Good Form is Talkative
A Good Form is Unbreakable
A Good Form is Fun
A Good Form is Concise
A Good Form is Well Formatted
A Good Form is Easy to Scan
A Good Form is Easy to Navigate

Thanks!

@shortdiv

https://codepen.io/collection/XgWrxd/

https://github.com/shortdiv/yanny-ya-hear-laurel